APPLICATION FOR UNITED STATES LETTERS PATENT

INVENTORS: Jeong Il BANG

TITLE: METHOD FOR DEBUGGING IN APPLICATION PROGRAM AND APPARATUS THEREOF

ATTORNEYS: FLESHNER & KIM, LLP
&
ADDRESS: P. O. Box 221200
Chantilly, VA 20153-1200

DOCKET NO.: HI-0066

METHOD FOR DEBUGGING IN APPLICATION PROGRAM AND APPARATUS THEREOF

BACKGROUND OF THE INVENTION

1. Field of the Invention

[1] The present invention relates to a method for debugging in an application program, and an apparatus thereof.

2. Background of the Related Art

[2] One typical method for debugging in an application program is to ban users from writing into a memory. That is to say, the related art application program uses a memory management unit (MMC) to separately protect on each page. In general, the related art application program has been used for protecting a code region.

[3] Fig. 1 diagrammatically illustrates a page descriptor used in a MMC in a related art for protecting memory. As shown in Fig. 1, when an error occurs in the application program, the MMC inside of a central processing unit (CPU) takes the responsibility of protecting the memory from the error and managing the subject memory. As shown in Fig. 1, depending on the condition whether a region is manifested as write-protected or not using a 'W' data/descriptor among other page descriptors, the MMC can designate write-protected by '1' or '0' in the 'W' data. In other words, if the MMC intends to designate a certain page region for saving as a write-protect region, the

MMC can designate the W data as '1'. In contrast, if the MMC intends to save the page region, it can designate the W data as '0'.

[4] Therefore, the related art method for protecting memory uses the MMC enabled to protect a write on each page. However, the related art method for protecting memory has various disadvantages. The write protection was not applicable to many application programs. Further, since the application programs have different stacks and are variable among one another, when an error occurred, the related art MMC was neither sufficient nor helpful to determine which application program had the error.

[5] The above references are incorporated by reference herein where appropriate for appropriate teachings of additional or alternative details, features and/or technical background.

SUMMARY OF THE INVENTION

[6] An object of the invention is to solve at least the above problems and/or disadvantages and to provide at least the advantages described hereinafter.

[7] Another object of the present invention is to provide a method for debugging in an application program and an apparatus for examining whether each application program violates a region assigned to each task.

[8] Another object of the present invention is to provide a method for debugging in an application program and an apparatus for examining whether each application program violates a region assigned to each task using comparison logic.

[9] Another object of the present invention is to provide a method for debugging in an application program and an apparatus thereof, using an interrupt signal.

[10] To achieve at least the above objects in a whole or in part, there is provided a method for debugging in an application program according to the present invention that includes writing information on a task to be performed; checking whether the task is performed in a designated region; and generating an interrupt signal, if the task is performed in another region instead of the designated region for the task.

[11] One method can further include latching a data signal corresponding to the written information on the task, and outputting a task signal corresponding to the task identifier that is identified based on the latched data signal.

[12] To further achieve the above objects in a whole or in part, there is provided a method for debugging in an application program, the method including (a) outputting a task signal corresponding to a task identifier that is identified based on a data signal corresponding to the task identifier, (b) checking an operation region of a task that is accessed based on an access of data, (c) judging whether the task is performed in a designated region based on an address signal corresponding to a result of the checking, and

(d) generating an interrupt signal when the task is not performed in the designated region as a result of the judging.

[13] To further achieve the above objects in a whole or in part, there is provided a apparatus for debugging in an application program, the apparatus including first control device for writing a task identifier provided for each task to be performed, for generating a data signal corresponding to the task identifier, and for activating a selected task, task checking device for outputting a task signal corresponding to the task identifier that is identified based on the data signal, and for generating an interrupt signal according to a determination whether a current task is performed in a designated region, and storage device for writing the task identifier provided by the first control device, and for assigning an operation region to each task.

[14] Additional advantages, objects, and features of the invention will be set forth in part in the description which follows and in part will become apparent to those having ordinary skill in the art upon examination of the following or may be learned from practice of the invention. The objects and advantages of the invention may be realized and attained as particularly pointed out in the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[15] The invention will be described in detail with reference to the following drawings in which like reference numerals refer to like elements wherein:

[16] Fig. 1 is a diagram that illustrates a page descriptor used in a related art memory management unit for protecting memory;

[17] Fig. 2 is a block diagram illustrating a preferred embodiment of an apparatus for debugging in an application program in accordance with the present invention;

[18] Fig. 3 is a block diagram illustrating a task testing unit in accordance with a preferred embodiment of the present invention;

[19] Fig. 4 is a block diagram illustrating a task identifying unit in accordance with a preferred embodiment of the present invention;

[20] Fig. 5 is a logic circuit diagram that shows a preferred embodiment of a comparing unit in accordance with the present invention; and

[21] Fig. 6 is a diagram that shows a flow chart of a preferred embodiment of a method for debugging in an application program in accordance with the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[22] Preferred embodiments of the present invention, examples of which are illustrated in the accompanying drawings will now be described. Fig. 2 is a block diagram illustrating a preferred embodiment of an apparatus for debugging in an application program according to the present invention. In preferred embodiments according to the present invention, application programs have to be programmed to be performed only

in corresponding pre-allocated areas. Moreover, operating systems, being activated by a central processing unit (CPU), are capable of managing the entire system and, if necessary, to operate at least one of the application programs in an operation region according to a designated procedure. As described below, one of the application programs can be designated as a task. Each task is also given a certain or prescribed region (e.g., in memory) assigned in advance for the task to be activated in the designated region only.

[23] As shown in Fig. 2, the preferred embodiment of an apparatus for debugging includes a central processing unit 10, a task testing unit 40, a memory controlling unit 30, and a memory 20. In addition, in order to perform the task, the central processing unit 10 preferably sends a task ID enable signal out to the task testing unit 40, and generates a data signal corresponding to the task ID, simultaneously. The data signal is a binary combination that can determine or identify the task ID. For example, if the task ID is 3, the data signal can be generated as '0011' from the most significant bits of a data bus. On the other hand, the central processing unit 10 can call the task corresponding to the task ID into the operation region, and proceed to activate the task.

[24] The task testing unit 40 will now be described in additional detail with reference to Fig. 3. As shown in Fig. 3, the task testing unit 40 preferably includes a task identifying unit 42 and at least one of task comparing units 1, 2, ... N (e.g., 48a through 48n). The task comparing units 48a-48n are preferably available equal in number to the tasks. The task identifying unit 42, as illustrated in Fig. 4, preferably includes a latching

unit 44 for latching the data signal on the basis of the task ID enable signal generated by the central processing unit 10, and a decoding unit 46 for outputting a task signal corresponding to the task ID that is identified based on the latched data signal. When the data signal for the next task is not yet received, the latching unit 44 preferably keeps outputting the present data signal and sends the present data signal to the decoding unit 46. The decoding unit 46 decodes the outputted data signal from the latching unit 44, and outputs a task signal. The task signal may be a signal inputted to a task comparing unit (e.g., 48a-48n) in order to perform a error testing of a task. Therefore, the decoding unit 46 can identify the corresponding task based on the task signal. Suppose that the task signal is '0011'. Then, the task signal can be inputted in to a corresponding one of the task comparing units, i.e., 48c, among other task comparing units (48a through 48n). However, the present invention is not intended to be so limited.

[25] On the other hand, the task testing unit 40, if the task is being currently performed, can preferably check whether the subject task is being performed in a designated or prescribed region or not. That is to say, the task testing unit 40 checks or tests the operation region for the task that is currently being performed, and according to the checking result, the task testing unit 40 generates an address signal. If the task is being performed in the designated region, the task testing unit 40 preferably does not generate the address signal. In contrast, if the task is not being performed in the designated region, the task testing unit 40 preferably generates the address signal, which

is later inputted in the corresponding task comparing unit 48a-48e, such as 48c, as described above. Besides the task testing unit 40, an additional separate unit (not shown) can also be provided to judge whether the task is being performed in the designated region or not in another preferred embodiment.

[26] As shown in Fig. 3, the task signal can be inputted into one of the task comparing units (e.g., 48c), and that task comparing unit can output a grant signal according to the address signal inputted based on the task signal. As shown in Fig. 3, instead of the grant signal, a write signal (WR) that is generated from the central processing unit 10 can be used instead. If the write signal (WR) is a prescribed value such as '1', the write signal (WR) is a read signal, and if the write signal (WR) is set to '0', the write signal (WR) is a write signal. However, it should be noted that the grant signal output is dependent or preferably totally dependent on the address signal. The address signal is preferably a signal for indicating whether a task is performed in the designated area or not.

[27] A preferred embodiment of a task comparing unit is illustrated in Fig. 5. As shown in Fig. 5, the task comparing unit (e.g., 48c) includes a OR-NOT gate 51 for applying the OR-NOT operation to the address signal, and an AND gate 53 that receives an output signal of the OR-NOT gate 51 and generates the grant signal based on the inputted task signal and the write signal (WR). Therefore, the task comparing unit (e.g., 48c) can generate the grant signal having prescribed values such as '0' or '1', in accordance

with the consequence of the address signal and based on both the task signal and the write signal (WR). At this time, if the grant signal '0' is generated, which preferably means that the task is not performed in the designated region, an interrupt signal can be generated.

[28] As shown in Fig. 2, when the interrupt signal is generated, the memory controlling unit 30 can send the interrupt signal to the central processing unit 10. In addition, the memory controlling unit 30 preferably outputs a control signal based on the interrupt signal in order to control the memory 20. The central processing unit 10, based on the interrupt signal, checks the ID of the task that is currently being performed, and writes the information about the task ID so that the corresponding task (e.g., currently being performed) can be debugged. The memory 20 preferably writes the task ID provided by the central processing unit 10, and assigns an operation region for each task.

[29] As described above, the preferred embodiment of the apparatus for debugging in an application program according to the present invention provides a checking process to find out whether the current task is being performed in the designated region. However, it should be also noticed that the same preferred embodiment of the present invention can be applied to another checking process to find out whether the data, which have been calculated by the task currently being performed, are being written properly in its designated region.

[30] A preferred embodiment of a method for debugging in an application program in accordance with the present invention will now be described. Fig. 6 is a

diagram that shows a flow chart for the preferred embodiment of a method for debugging in an application program. As shown in Fig. 6, the preferred embodiment of a method for debugging can be used, for example, in the apparatus of Fig. 2. After a process starts in Fig. 6, control continues to step S61 where once a board is booted and the operating system is activated, the central processing unit 10 writes the task ID of the task that needs to be performed in the memory 20. From step S61 control continues to step S63.

[31] In step S63, the decoder (not shown) decodes the address and control signal and sends a task ID enable signal to the latching unit 44 so that the task can be preferably immediately performed. The latching unit 44 latches the data signal corresponding to the task ID, and sends the data signal to the decoding unit 46. Then, the decoding unit 46 decodes the data signal, generating a task signal, and finally outputs the generated signal to one (or more) of the comparing units. From step S63 control continues to step S65 where the central processing unit 10 preferably concurrently calls the task corresponding to the task ID into the operation region, and performs the task. The operation region preferably means a pre-designated region by the task. If there are a number of tasks to be performed, an operation region can be designated for each task in advance.

[32] From step S65, control continues to step S67 where during the performance of the task, it is determined whether the task is being duly performed in the designated region. In other words, as the task is performed, it is necessary to ascertain the operation region of the task. If the operation of the current task is the pre-designated region, the

address signal is not generated and no error occurs for that task and control jumps to step S73.

[33] In contrast, if it is determined in step S67 that the operation region of the task is not the pre-designated region for that current or particular task, the address signal is generated and the address signal is inputted into at least one task comparing unit. The task comparing unit (e.g., 48c) can generate the grant signal '0', based on the generated address signal. Then, upon the generation of the grant signal '0', the interrupt signal can be generated in consequence in step S69. If the generated grant signal is '1' instead of '0', no interrupt signal is preferably generated.

[34] The memory controlling unit 30 not only preferably sends the interrupt signal to the central processing unit 10, but also outputs the control signal for controlling the memory based on the interrupt signal. From step S69 control continues to step S71. In step S71, the central processing unit 10 generates the information of the task that generated the interrupt signal by performing an interrupt service routine based on interrupt signal. Preferably, the error information of the task generated the interrupt signal is stored in the memory 20 through performance of the interrupt service routine, and a user ascertains whether any application program is generated in error using the error information of the task.

[35] As described above, if it turns out that the operation region of the task is the pre-designated region for the task in step S67, the central processing unit 10 certifies the

information of a task switching based on another interrupt signal generated in other parts of the apparatus of the present invention in step S73. If it is determined in step S73 that the task switching occurred, control jumps back to step S61. On the other hand, if the task switching is not generated, control jumps back to step S65 and the central processing unit 10 continuously executes the application program corresponding to the task. Task switching indicates a situation where the next task is performed in the middle of performing the current task.

[36] As described above, preferred embodiments of a method and apparatus for debugging in an application program according to the present invention have various advantages. The preferred embodiments for an apparatus and method for debugging in an application program according to the present invention can reduce or solve a problem that the application program has an error occur, the error is recorded accurately. The preferred embodiments can check whether the application program is duly performed in the designated region and control the application program to be performed in the designated region based on a result of the checking. A preferred embodiment of an apparatus for debugging of the present invention including a comparison logic circuit that is low cost, small sized and more efficiently reaches accurate diagnosis.

[37] The foregoing embodiments and advantages are merely exemplary and are not to be construed as limiting the present invention. The present teaching can be readily applied to other types of apparatuses. The description of the present invention is intended

to be illustrative, and not to limit the scope of the claims. Many alternatives, modifications, and variations will be apparent to those skilled in the art. In the claims, means-plus-function clauses are intended to cover the structures described herein as performing the recited function and not only structural equivalents but also equivalent structures.